基于 ECO 与 Ajax 相结合的用户注册验证

庄严,倪子伟

(厦门大学 计算机科学系,福建 厦门 361000)

摘要:ECO 是 Borland/CodeGear 在.NET 平台下实现出来的,根据模型驱动开发(Model Driven Development, MDD)为核心发展出来的软件架构;而 Ajax 作为 Web 开发的热点,在提高数据交互,改善用户体验等方面发挥着重要作用。本文通过用户注册验证的设计,说明了通过 Web 服务,ECO 和 Ajax 二者之间能实现有机的结合,从而让开发人员真正领略到高效的开发能力,同时带来更好的用户体验。

关键词: ECO; Ajax; Web Service; .NET

中图分类号:TP311　　**文献标识码**:A　　**文章编号**:1009-3044(2008)14-20848-04

User Registration Verification Based on the Combination of ECO and Ajax

ZHUANG Yan, NI Zi-wei

(Xiamen University Department of Computer Science, Xiamen 361000, China)

Abstract: ECO is Borland/CodeGear. NET platform to achieve out, according to model-driven development as the core software architecture developed while Ajax as a Web development of the hot spots, while enhancing the interactive data to improve the user experience, and played an important role. In this paper, the design of user registration verify that through Web services, Ajax between ECO and to achieve the combination of organic, thus allowing developers to truly appreciate the efficient development capabilities, and at the same time will enhance the user experience.

Key words: ECO; Ajax; Web Service; .NET

1 引言

随着高生产力和高品质软件的要求标准不断的提高,‘简化与快速开发’已成为软件开发人员追求的重要目标。在‘快速开发”的同时,提供丰富的、可交互的和个性化的用户体验,对提高 Web 应用程序开发的品质和效率,具有重要的现实意义。

2 ECO 与 Ajax 相结合的模式

2.1 ECO 开发架构

ECO(Enterprise Core Objects)是根据模型驱动开发(Model Driven Development,MDD)为核心发展出来的技术,并且结合 OR Mapping、图形用户界面绑定、对象服务构架以及许多其他丰富的功能而形成的,在.NET 平台下实现出来的软件工程。

2.1.1 模型驱动开发(Model Driven Development,MDD)

MDD 的基本构想是开发人员应该专注于设计良好的业务逻辑模型,并且把业务逻辑撰写在模型中。一旦模型设计完成之后,MDD 即可提供工具自动地把业务逻辑模型对映到开发人员使用的技术中。MDD 提供了使用模型来直接引导理解系统、设计、建立、部署、动作、维护和修改系统的流程。在这整个开发过程中,都是以模型来驱动的。一个 ECO 系统的开发从建立模型、设计和开发模型,到撰写 Delphi/C# 程序代码都与系统模型脱离不了关系。

2.1.2 UML 和 OCL

UML(Unified Modeling Language)和 OCL(Object Constraint Language)都是由 OMG(Object Management Group)定义的标准,也都是组成 MDD 标准的子标准之一,对 MDD 的成功具有关键性的作用。ECO 开发中,使用 UML 来专注于建立业务的平台无关模型(Platform Independent Model,PIM),使用 OCL 来选择和处理业务逻辑对象和业务逻辑。

在 UML2.0 的标准规范中,OCL 是作为附加的语言(Add-on Language)来进行面向对象分析/设计之用的。由于 OCL 是附加的 UML 机制,所以被用来撰写业务逻辑模型中的业务规则、限制条件以及定义查询等工作之用。OCL 又是一门形式语言,因此使用 OCL 来定义、撰写这些模型相关的信息的好处是既清楚又不会造成模糊不清的定义。OCL 可以取代 SQL 在 ECO 应用系统中让开发人员处理的业务逻辑对象,也可使用在 ECO 类图中定义的业务逻辑,这样,开发人员就无须再使用 Delphi/C# 程序代码重复撰写这些业务逻辑,可以说,使用 ECO 来开发应用系统,OCL 是除了 Delphi/C# 语言之外第二个最重要的语言。

2.1.3 使用 ASP.NET 之 WEB 架构的 ECO 应用程序

ECO 的开发方式是让开发人员设计完善的用户需求业务逻辑模型,不仅定义了类、方法和特性,更重要的是它详细地定义了一个应用系统中类之间的关系,这样一来,开发人员只需要遵照模型的定义就可以专心地撰写程序代码,因此,可以简单地说 ECO 能够达到‘模型即程序,程序即模型’的地步。

ECO 提供了强大的 ASP.NET 开发能力,它能够和.NET 内建的 UI 组件搭配在一起,并且重新整理高阶的服务在一组易于使用的接口中,这样一来,不但可以让开发者非常方便上手,也符合目前面向对象框架改为使用接口导向(Interface Oriented)的设计架构。在这种 ECO ASP.NET 架构中,除了能够使用所有 ASP.NET 提供的各种组件和服务之外,也能够使用所有 ECO 的功能和服务。ASP.NET 应用程序同样通过 EcoSpace 执行我们在设计时期设计的业务逻辑模型,然而,由于 ASP.NET 应用程序会有许多客户端同时存

收稿日期:2008-03-19

作者简介:庄严(1970-),男,福建惠安人,厦门大学计算机科学系学生;倪子伟,男,厦门大学计算机科学系副教授。

取使用,因此 ASP.NET 应用程序需要和后台的数据库连接,并需要使用 ECO 持久化映像同步器(PMapper Synchronizer)来同步每一个客户端对于对象的修改,以保证后台数据库之中数据的一致性。执行架构如图 1。

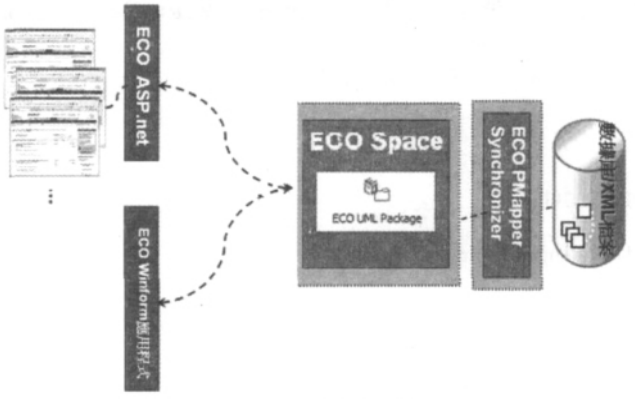

图 1　ECO 执行架构

2.2 AjaxPro.NET 框架

AJAX 是 Asynchronous JavaScript and XML(异步 JavaScript 和 XML)的缩写,综合运用 Javascript、XHTML 和 CSS、DOM、XML 和 XSTL、XMLHttpRequest 的技术,AJAX 的原则是"按需取数据",可以最大程度的减少冗余请求和响应对服务器造成的负担,它使用 XMLHTTP 对象发送请求并得到服务器响应,在不重新载入整个页面的情况下用 Javascript 操作 DOM 实现页面局部更新,这种更新是瞬间的,从而带给用户较好的体验。

随着 Ajax 技术在 WEB 程序中的应用,Ajax 框架(或称 Ajax 库)有如雨后春笋般令开发人员眼花缭乱,但是面对如此之多的 Ajax 框架,如何选择适合自己项目的 Ajax 框架,特别是在和 ECO 相结合的应用系统中,显得更为重要。

AjaxPro.NET 是一个优秀的 Ajax 框架,它做为 Ajax 技术在.Net 框架下的实现,作者 Michael Schwarz 采取了一种封装效果相当棒的技术:将客户端处理 XML、事件调用方式都封装在 2 个 Javascript 文件中(AjaxPro.prototype.js 和 AjaxPro.core.js),同时将这 2 个重要文件以资源的形式编译于 dll 中,在处理客户端请求其自定义的.ashx 文件时,返回这两个文件。在实际应用中只要添加其 DLL 引用并进行简单的配置,就可以非常方便的在客户端直接调用服务端方法,实现验证目的。

2.3 Ajax 通过 WebService 整合 ECO 系统

ECO 功能强大,而且非常符合开发习惯,当使用 ECO 开发应用程序时,开发人员可以专注于设计要解决的问题模型,一旦这个模型定义、开发完毕之后,整个应用程序就几乎已经完成了大半的工作,因为 ECO 在应用程序运行时会执行开发人员设计的模型来解决客户的问题。这样开发人员可以避免花费很多的开发时间于琐碎的细节之中,而能集中精力于设计完善的业务逻辑。

Ajax 作为构建 Web 应用的一种方法,它带给访问者完全不同的浏览感受,为客户端提供了强大的 GUI 能力,通过 ECO 和 AJAX 的有机结合,开发人员可以在 WEB 应用程序中使用完善的业务逻辑,并将良好的用户体验展现在使用者面前,那么在 ECO 与 Ajax 相结合的模式下,服务器端的开发应该是什么样子,应该用什么样的框架支持这种"只向浏览器返回数据"的服务请求,其实这时很需要一个全新概念的服务器端架构,本人认为这种架构应该基于 WebService。

Web Service 主要是为了使原来各孤立的站点之间的信息能够相互通信、共享而提出的一种接口。Web Service 所使用的是 Internet 上统一、开放的标准,如 HTTP、XML、SOAP(简单对象访问协议)、WSDL 等,所以 Web Service 可以在任何支持这些标准的环境(Windows,Linux)中使用。Webservice 的一个最基本的目的就是提供在各个不同平台的不同应用系统的协同工作能力。

通过 Web 服务,Ajax 技术可以与 ECO 应用程序很好进结合在一起,二者可通过如图 2 的模式进行相结合。

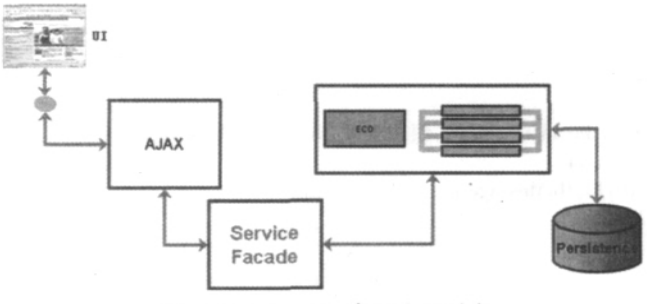

图 2　ECO 和 Ajax 相结合的模式

3 结合 ECO 与 Ajax 的用户注册验证设计与实现

在各种网络服务中,新用户注册随处可见,新用户进行注册时,系统应判断注册的用户名是否已存在,ECO 系统中如何在不重新刷新页面的情况下进行验证,就须通过使用 Web Service 方法。Ajax 与 ECO 相结合下的用户注册验证的实现主要有以下几个步骤:

(1)首先下载 AjaxPro.NET 组件。并将 AjaxPro.2.dll 引用到项目中;

(2)在 Web.config 文件中配置 AjaxPro 框架;

```
<system.web>
<httpHandlers>
<add verb="*" path="*.ashx" type="AjaxPro.AjaxHandlerFactory,AjaxPro.2"/>
```

```
</httpHandlers>
</system.web>
```

(3)使用 ECO 设计器界面以可视化的方式设计注册用户类(Register 类有:UserName、Password、Email 等注册需要的属性),然后通过 OR Mapping 在连接的数据库中产生必要的数据结构;

(4)创建 ECO WebService,在服务器端撰写 Web 服务方法;

```
unit WebService1;
……
[WebMethod]
function GetUserCount(Uname:string): integer;
……
function TWebService1.GetUserCount(Uname:string): integer;
var
OclService: IOclService;
ResultElement: IElement;
oclstr:string;
begin
OclService := EcoSpace.GetEcoService<IOclService>();
oclstr:='Register.allinstances->select(UserName='''+Uname+''') ';
ResultElement := OclService.Evaluate(oclstr);
Result := ResultElement.GetAsCollection.Count;
DoneWithEcoSpace;
end;
```

将调试后的 Web 服务方法通过“Web Reference”添加引用到调用的项目中。

(5)在 ECO ASP.NET Forms 中注册使用 Ajax 框架,同时加入相关的 ECO 组件,设计调用 WebService 的方法;

```
public
function IsNameExist(sUname:string):boolean;
function IsUserExist(Uname:string):boolean;
[AjaxPro.AjaxMethod]
function GetReturnInfo(s:string): string; //声明客户端 JavaScript 可呼叫的方法
[AjaxPro.AjaxMethod]
function GetInfo : string; //声明客户端 JavaScript 可呼叫的方法
……
procedure TRegist.Page_Load(sender: System.Object; e: System.EventArgs);
begin
// TODO: Put user code to initialize the page here
Utility.RegisterTypeForAjax(typeof(Default.TRegist));
end;
function TRegist.IsUserExist(Uname: string): boolean;   //调用 WebService
var
ws:TWebService1;
begin
ws:=TWebService1.Create;
if ws.GetUserCount(Uname)>0 then Result:=true else Result:=false;
end;
function TRegist.GetReturnInfo(s: string): string;
begin
if(IsUserExist(s)) then Result:='2';
end;
```

(6)撰写客户端访问 ASPX 页面的 JavaScript 代码,使用 XMLHttp 呼叫 Ajax 方法。

```
function IsUsernameExist_callback(res)
{var msg = document.getElementById('lblMessage');
var bRet = res.value;
if (bRet == "2")
{msg.innerHTML = '用户名已存在! ';
msg.style.color = ' red ';}
else
{msg.innerHTML = '用户名可以使用! ';
msg.style.color = 'green'}
}
</script>
……
<body>
<form runat="server">
<asp:TextBox id="txtUserName" runat="server" onkeyup="VerifyUsername(this.value)"></asp:TextBox>
<asp:Label id="lblMessage" runat="server"></asp:Label>
```

```
</form>
</body>
```

4 结束语

本用户注册验证程序在 RAD Studio 2007 下调试成功。本文通过这一常用实例的实现,提出一个在.NET 平台下快速开发 Web 程序的基本模式:Ajax 技术可以通过 WebService 和 ECO 应用程序进行结合,充分实现“快速开发”与“良好用户体验”,较好地完成 Web 应用软件的开发。

参考文献:

[1] 李维. Delphi MDA/DDA 程序设计——使用 ECO[M]. 电子工业出版社,2007.
[2] Xavier Pacheco. Delphi for.NET Developer's Guide[M]. 机械工业出版社,2005.
[3] 柯自聪. Ajax 开发精要——概念、案例与框架[M]. 电子工业出版社,2006.
[4] 蔡宏. Delphi 2006 for NET 开发技术原理与实践教程[M]. 电子工业出版社,2007.
[5] 赵建华. MDA 与可执行 UML[M]. 机械工业出版社,2006.

(上接第 842 页)

消防信息管理系统采用面向服务的体系结构,由 Web Service 作为中间层负责身份验证和数据处理,并与后台数据库服务器交互,来检索数据或验证用户身份。

4.2 身份验证 Web Service

身份验证 Web Service 的执行过程是:用户提供的用户名和密码,通过身份验证 Web Service 递交给数据库,数据库使用一个存储过程验证用户名和密码,如果验证成功,返回嵌套了用户 ID 的唯一加密票。如果验证用户名和密码失败,则不返回任何内容。

4.3 数据 Web Service

数据 Web Service 提供了客户端应用程序用以检索和更改数据的功能,并且提供了身份验证服务,能够验证用户的每个请求。数据 Web Service 的每种公共 Web 方法都要提供调用者的身份加密票。在返回任何数据之前,在加密票缓存中检查票是否存在。如果存在,说明在最近某端段时间内对用户名和密码进行了验证;否则票将无效或过期,如果票无效或过期,会从票中提取用户 ID,并执行身份验证过程重新验证用户 ID 后,再向下执行。

5 客户端设计与实现

5.1 数据层组件

数据层类 DataLayer 处于用户界面和数据 Web Service 中间,用户界面的数据请求会通过数据层类间接调用。数据层类相当于用户表示层与业务逻辑层之间的代理类,由它调用业务逻辑层功能。数据层类实现了程序界面与数据的松散耦合,系统与数据相关的操作,如数据检索、修改、身份验证等,都由数据层间接调用 Web Service 来完成。

5.2 数据冲突处理

数据冲突处理主要是处理并发,并发是多个用户访问相同数据的管理策略。并发主要分为开放式并发和保守式并发。本系统采用的是开放式并发。

当客户端尝试更新或删除数据库中的数据时,若这些数据自该客户端上次访问它们以来已被更改,或者根本不存在,就会发生数据冲突。通常可以通过引发错误或者简单地使用客户端版的记录重写数据库中的任何内容来处理。第一种方案会导致客户端的工作无效。第二种方案带来的风险是忽略和删除自从客户端上次检查数据库以来输入的重要数据。在该系统中对此问题引入了一个简单的解决方案,主要依靠.NET 框架中的 ADO.NET 库的 DataSet 对象中的功能。

5.3 脱机—联机工作模型

消防信息管理系统可以处于脱机或联机两种工作模式,工作模式的选择是由单击脱机/联机按钮来完成的。程序首先检查是否存在本地脱机数据文件和更改数据文件,如果这些文件存在,将加载它们到 DataSet 并以脱机模式运行。若这些文件不存在,程序将以联机模式运行,直接通过调用数据 Web Service 与数据库进行交互。

参考文献:

[1] Microsoft MSDN 技术文档.智能客户端定义.http://www.microsoft.com/china/msdn/developercenter/smartclient/SmartClientDefinition.aspx.
[2] 刘海波,钟志农,陈宏盛,等.智能客户端技术研究及应用[J].兵工自动化,2006(11).
[3] David Hill,Brenton Webster,Edward A.Jezierski. 智能客户端体系结构与设计指南.http://www.microsoft.com/china/msdn/library/architecture/architecture/architecturetopic/SCArchDeGuide/cover.mspx?mfr=true.